AIAA Aviation
13-17 June 2016, Washington, D.C.
16th AIAA Aviation Technology, Integration, and Operations Conference

A Mixed Integer Linear Programming Approach for Computing the Optimal Chance-Constrained Push Back Time Windows

William J. Coupe * and Dejan Milutinović †
Jack Baskin School of Engineering, University of California, Santa Cruz, CA 95064, USA
Waqar Malik ‡
University of California, Santa Cruz, NASA Ames Research Center, Moffett Field, CA 94035, USA
Yoon Jung §
NASA Ames Research Center, Moffett Field, CA 94035, USA

Due to the stochastic uncertainties in ramp area aircraft trajectories, the optimization of push back time windows has to account for randomness associated with the trajectories. This paper formulates a mixed integer linear program for computing the optimal chance-constrained aircraft push back time windows. The solutions are chance-constrained because they allow for a non-zero but bounded probability of conflicts among the sampled aircraft trajectories. Solutions of the mixed integer linear program are shown to be significantly impacted by the presence of even a few rare conflict points within an otherwise empty domain. By allowing for some conflict points inside the time windows, the solutions become much more attractive. The runtime of the mixed integer linear program is shown to be most influenced by a parameter within the objective function, the distribution of conflict points, and the number of conflict points that are allowed inside the time window. In order to reduce the runtime we introduce various cutting methods applied to the mixed integer linear program domain. Overall, the analysis shows that the cutting methods reduce both the runtime and standard deviation of the runtime for the mixed integer linear program.

I. Introduction

During periods of heavy surface traffic, the NASA Ames Spot and Runway Departure Advisor [1–7] (SARDA) directs departing aircraft to remain at the gate with their engines off, and when cleared, they can proceed straight from the gate to the departure runway queue *minimizing* slowing down or stopping for other traffic [2] and still meet their target movement area and take-off times. This technique has the capability of significantly reducing fuel burn and engine emissions.

A recent study [8] estimated as much as 18% of fuel consumption during taxi operations was due to stop-and-go activity. The study also concluded that under the assumption of 15 knots or greater speed for all unimpeded aircraft, there is the potential to reduce overall fuel consumption on the surface by at least 21%. In order to execute unimpeded surface trajectories, it is required that airports have the necessary tools to meet the assigned target movement area times. Although surface operations can be improved by adopting an optimal taxiway schedule, the execution ultimately depends on human controllers who control aircraft maneuvers in both ramp area and taxiways [9].

This research proposes a tool to aid ramp area controllers in meeting the scheduled target movement area times by computing the push back time window for each departing aircraft. The push back time window is defined by the range between the earliest feasible push back time and latest feasible push back time.

*AIAA Graduate Student Member, Graduate Student, Computer Engineering Department, UC Santa Cruz.
†Associate Professor, Computer Engineering Department, UC Santa Cruz
‡Research Scientist, University Affiliated Research Center, MS 210-8, Moffett Field, CA 94035.
§AIAA senior member and Aerospace Engineer, NASA Ames Research Center, MS 210-6, Moffett Field, CA 94035.

Initiating the push back within these bounds ensures there exists a feasible trajectory that arrives at the target movement area at the required time, which is defined by a higher level optimal taxiway scheduler.

The main contribution of the tool is to compute push back time windows that allow for aircraft to taxi unimpeded from their gate to the departure runway queue in the presence of other aircraft and trajectory uncertainties. This allows for ramp controllers to better manage surface traffic and reduce fuel consumption while scheduling the push back for ramp area aircraft. Ideally, the tool will be used for real-time decision making by controllers so the computational runtime becomes a critical component of the tool.

For relative target movement area schedules resulting in conflict free aircraft trajectories, computing the push back time windows are straightforward and can be estimated from the sampled trajectories. To compute the push back time windows we estimate the maximum trajectory duration and minimum trajectory duration for each aircraft, and then subtract from the scheduled target movement area time for the given aircraft. These computed times represent the start and finish of the feasible push back time window for each aircraft, respectively. When the schedule may lead to aircraft trajectory conflicts, an optimization procedure which solves for push back time windows in the presence of aircraft trajectory uncertainties is needed.

To account for the uncertainty of ramp area operations, we proposed a stochastic model of ramp area aircraft trajectories [10, 11]. The stochastic model was used to sample a large number of feasible ramp area aircraft trajectories. A feasible trajectory is any sampled trajectory from the stochastic model that arrives at the target movement area within a predefined range of heading angles. The set of feasible trajectories for each aircraft is sampled in the absence of other aircraft. Therefore, the set of trajectories represent the feasible ways in which the aircraft can push back from their gate and taxi to the target movement area unimpeded by other aircraft.

Using the sampled trajectories, we computed combinations of push back times that lead to conflict of trajectories between any two aircraft, defined by the time separation of aircraft at the target movement area. The conflict distributions are defined by a measure of conflict estimated from the ratio of conflicting trajectories to total trajectories for every relative target movement area schedule. The conflict distributions were used to compute conservative conflict separation constraints that were passed to a mixed integer linear program (MILP) [12–14] which incorporated the Spot and Runway Departure Advisor (SARDA) design approach.

Previously, we proposed a MILP approach to solve for the optimal combination of push back time windows [15]. The optimal combination of push back time windows was defined to maximize the minimum push back time window of a set of all aircraft being scheduled. The solutions were based on conflict points that represent combinations of push back times that result in aircraft trajectory conflicts. We used the idea that no conflict point should be a convex combination [16] of the points in time that define the start and finish of the push back time window. In one-dimension, a convex combination of two points lies in between the two points. The number of constraints that are passed to the MILP is a function of the number of conflict points and for every conflict point, we need five constraints.

This approach was conservative in nature because we solve for the combination of push back time windows that allow for *zero* conflict points inside. Given that we sample ramp area aircraft trajectories from a stochastic model, it is possible to sample trajectories and resulting conflict points that are extremely rare. These rare events can cut off otherwise feasible portions of the push back domain.

To address this, in this paper we propose a new MILP to solve for the optimal chance-constrained push back time windows. Chance-constrained programming is defined to maximize an objective function subject to constraints on variables that must be held at prescribed levels of probability [17]. The time windows are chance-constrained because they allow for a non-zero but bounded number of conflict points inside them. We find these solutions acceptable as the conflicts may be extremely rare. Furthermore, we do not expect that executing a schedule that leads to a sampled conflict will result in a conflict in real life. Pilots always have the option to slow down or stop along the route to avoid a loss of separation between aircraft. Therefore, we expect there to be a trade-off between the number of conflict points allowed inside the time window and the likelihood that pilots will have to slow down or stop aircraft to avoid a loss of separation.

This paper is organized as follows. In Section II we formulate the optimization problem. Next, in Section III we formulate a MILP approach to solve for the chance-constrained optimal time windows. This approach is similar in nature to our previous approach that solved for the time windows which allowed for zero points inside the time windows. Then in Section IV we analyze the solutions and runtime of the MILP and suggest ways to speed up the algorithm. In the last Section, we conclude with a discussion of our findings and provide directions for future work.

II. Problem Formulation

In this section we formulate the problem of computing the optimal chance-constrained ramp area aircraft push back time windows for each departing aircraft. The combination of push back time windows will be constrained to allow a non-zero but bounded number of conflict points inside the time windows. We begin by defining the variables and parameters that we use:

Symbol	Description
i	A family of departing trajectories originating from a single gate and characterized by a single left or right push back maneuver pattern
$\mathbb{D}$	The set of all possible departing aircraft push back maneuver patterns
t_i	The scheduled time for family i to arrive at the target movement area
t_i^S	The start of the computed push back sub-window for family i
t_i^F	The finish of the computed push back sub-window for family i
J	The objective function that is being maximized
t_i^{S0}	The start of the feasible push back time window for family i scheduled at $t_i = 0$
t_i^{F0}	The finish of the feasible push back time window for family i scheduled at $t_i = 0$
T_i	The set of all sampled trajectory duration data for departing trajectory family i
PB^i	The push back time of a single aircraft trajectory from family i
κ	A combination of push back times (conflict point) that lead to conflict among family i and j
M	A continuous variable representing the size of the minimum push back time window
$\delta_{\min}$	A parameter representing the minimum acceptable push back time window
ϵ	A parameter in the objective function J which influences the shape of the push back time windows
v_κ	A binary variable that is one if the conflict point κ is to be constrained outside the time window, otherwise zero
$z_{\kappa 1}$	A binary variable that is one if the time window is to be constrained below the conflict point κ, otherwise zero
$z_{\kappa 2}$	A binary variable that is one if the time window is to be constrained above the conflict point κ, otherwise zero
$z_{\kappa 3}$	A binary variable that is one if the time window is to be constrained left of the conflict point κ, otherwise zero
$z_{\kappa 4}$	A binary variable that is one if the time window is to be constrained right of the conflict point κ, otherwise zero
N	A constant representing the total number of conflict points κ
p	A constant representing the total number of conflict points κ allowed inside the time windows
S	A constant that is used in the linearization of nonlinear constraints
$\hat{\kappa}$	A conflict point κ that has been allowed inside the time window

A departing aircraft is parked at the gate and scheduled to arrive at the target movement area. In this paper, we assume the target movement area time is provided from a higher level taxiway scheduler. Upon receiving the push back clearance, a tug (operated by ground crew) pushes back the aircraft from the gate. At the end of the push back procedure, the aircraft stops and the tug disengages. This stop period lasts for some time during which the pilot goes through a checklist and then starts the aircraft engine(s). When ready the pilot requests taxi approval, and after the approval, taxies the aircraft until arriving at the target movement area. The target movement area is the point in space where departing aircraft transition from the ramp area into the Federal Aviation Administration (FAA) controlled taxiway. During the departure maneuvers the duration of the trajectory, the transitions over the motion phases, and the trajectory path are determined by human operators and are considered to be stochastic in nature.

Modeling ramp area aircraft departure maneuvers as stochastic processes, we sample a large number of departing trajectories from the stochastic model [10,11]. The sampled trajectories define a family i of feasible trajectories that originate from a single gate and arrive at the target movement area at the scheduled time

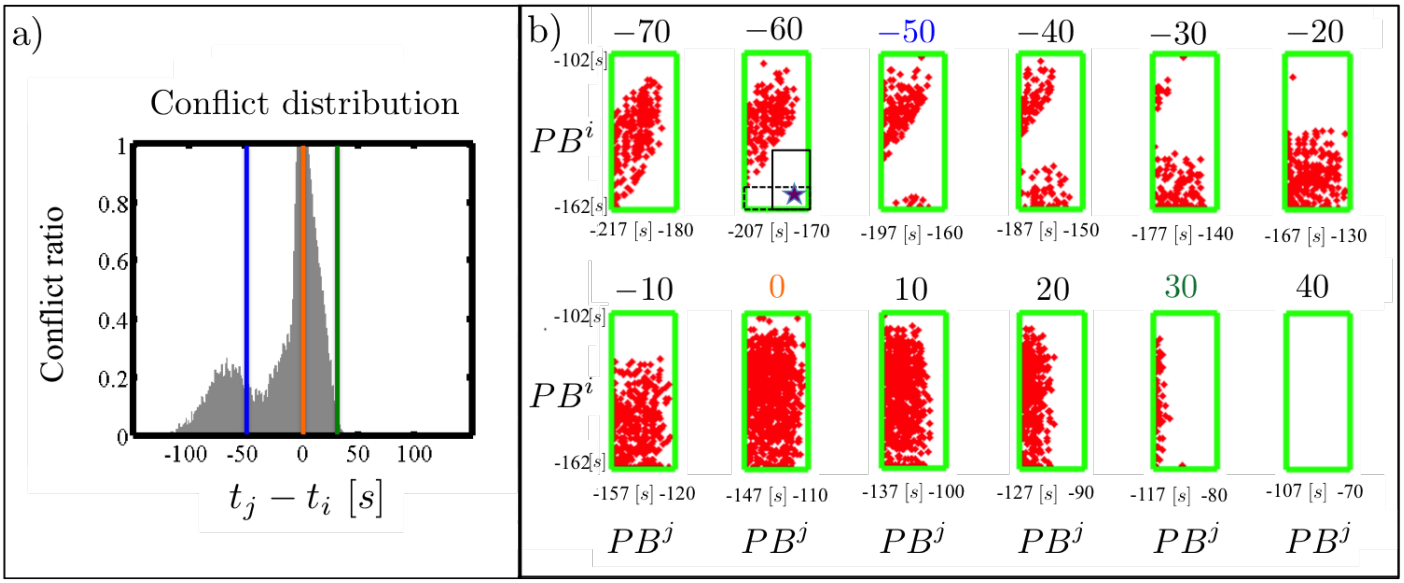

Figure 1. a) DFW conflict distribution with select cross sections colored. b) Plot of combinations of push back times (red points) resulting in conflicts between aircraft i and i for schedules ranging from $t_j - t_i = -70$ to $t_j - t_i = 40$ at a resolution of 10 [s]. The y-axis represents the push back time of aircraft i and the x-axis represents the push back time of aircraft j. If we do not account for conflicts the green rectangle represents the feasible push back domain. For the schedule $t_j - t_i = -60$ two feasible push back sub-windows are plotted in black solid and dotted lines.

t_i. We sample families of trajectories for each unique push back maneuver pattern $i \in \mathbb{D}$ where the set $\mathbb{D}$ denotes a set of all possible push back maneuver patterns, i.e. a left or right push back from the gate.

Using the family of trajectories i and j, we generate a conflict ratio defined by their relative schedule $t_j - t_i$ [10,11]. A conflict ratio is estimated by fixing the relative schedule of the two families of trajectories and computing the ratio of conflicting trajectories to the total number of trajectories. A conflict is characterized by individual trajectories from the families i and j coming into close spatial proximity along their route. The conflict distribution is estimated by computing a conflict ratio at every whole second, as shown in Fig. 1a. In the figure the y-axis represents the ratio of conflicting trajectories to conflict-free trajectories and the x-axis represents the relative target movement area schedule $t_j - t_i$.

For departing trajectory family i with scheduled target movement area time $t_i = 0$, the start of the feasible push back time window is defined by $t_i^{S0} = -\max_i(T_i)$ and the finish of the feasible push back time window is defined by $t_i^{F0} = -\min_i(T_i)$. The variable T_i is the set of all trajectory duration data for family i that is sampled from the stochastic model. For any given relative schedule, the earliest and latest feasible push back times define the green edges of the rectangle that are seen in Fig. 1b. The distribution in trajectory duration is estimated from the robot experiment data which is directly influenced by the human operator [10, 11]. We use data from a scaled down robot experiment of the ramp area because trajectory data are not readily available mainly due to the lack of surveillance equipment in the ramp area.

For the scheduled target movement area time differences that have a non-zero ratio of conflicts, we can store and plot the combination of push back times that lead to conflicts. In Fig. 1b the vertical axis represents the push back time of an individual trajectory from family i, PB^i, and the horizontal axis represents the push back time of and individual trajectory from family j, PB^j. In Fig. 1a we color select cross sections of the conflict distribution to demonstrate the relationship between the ratio of conflicts (Fig. 1a) defined by the difference between their scheduled target movement area times and the set of red conflict points (Fig. 1b) defined by the combination of push back times that lead to conflicts for the given target movement area schedule.

The combinations of push back times that lead to conflicts among individual trajectories from family i and family j are plotted (see Fig. 1b) in 10s increments for the target movement area time differences ranging from $t_j - t_i = -70$ to $t_j - t_i = 40$. Given that we are interested in the relative scheduled difference between the two families, we fix the target movement area time of family i such that $t_i = 0$, and the relative schedule difference is defined by the target movement area time of family j. Associated with each difference in scheduled target movement area time, e.g., $t_j = -70$, is a green rectangle that is defined by the earliest

and latest feasible push back times for each family such that the target movement area time schedule is satisfied. Thus, in order to satisfy the target movement area time $t_i = 0$, any individual trajectory from family i must push back within the window $PB^i \in [-162, -102]$ and to satisfy the target movement area time $t_j = -70$ any individual trajectory from family j must push back within $PB^j \in [-217, -180]$. For -70 there is a set of combination of push back times that lead to conflicts. These combinations are labeled as red conflict points $\kappa = (PB^j, PB^i)$ within the green rectangle (see Fig. 1b).

Consider the distribution of red conflict points for the scheduled target movement area time difference of -60 shown in Fig. 1b. We observe that in the bottom right of the green rectangle there is a large area that does not contain any red conflict points, only the purple star conflict point. If we restrict aircraft trajectory families i and j to push back within the lower right corner of the green rectangle, then with high probability the families of trajectories will be conflict free. Two potential solutions are shown in the lower right of the green feasible domain where the first solution is shown with a solid black line and the second solution with a dotted black line.

Among all possible solutions, we define the optimal combination of push back time windows to be the combination where we maximize the minimum time window. By maximizing the minimum time window we compute solutions that ensure any single aircraft's push back time window is not excessively reduced in size to accommodate other aircraft. The objective function for aircraft trajectory families i and j is defined as

$$\max_{t_i^S, t_i^F, t_j^S, t_j^F} \quad J := \left[(1-\epsilon)M + \epsilon(t_i^F - t_i^S + t_j^F - t_j^S)\right] \tag{1}$$

where M is a continuous value representing the size of the minimum push back time window among both aircraft i and j and $\epsilon \in [0, 1]$. The value M is not known *a priori* and is a variable in the program which is solved for. In order for the problem to be well defined, we include the variable M in the constraints to ensure that each individual time window is greater than or equal to the value M.

The cost function J is a function of the four variables t_i^S, t_i^F, t_j^S, t_j^F which represent the start and finish of the push back sub-window for aircraft trajectory families i and j, respectively. The four variables together define a combination of push back sub-windows such as the windows labeled with the solid (dotted) black lines in Fig. 1b. The selection of parameter $\epsilon = 0$ defines the objective function to maximize the minimum push back time window (min edge of time window) and the selection $\epsilon = 1$ defines the objective function to maximize the summation of push back time windows (perimeter of time window).

The optimization problem is subject to the constraints that no more than p conflict points can be contained within the optimal combination of push back sub-windows. For any given relative target movement area schedule, at a resolution of 1[s], we consider computing the optimal combination of push back sub-windows as defined above.

III. MILP for Computing Optimal Chance-Constrained Push Back Windows

Here we provide the mathematical formulation for the constraints of the optimization problem. For departing aircraft trajectory families $i, j \in \mathbb{D}$ we introduce the two constraints

$$t_i^F - t_i^S - M \geq 0 \tag{2}$$
$$t_j^F - t_j^S - M \geq 0 \tag{3}$$

that ensure the push back time window for aircraft trajectory family i and the push back time window for aircraft trajectory family j are both greater than the size of the minimum time window M. We note that the value M is not a fixed value, but a continuous variable that we pass to the solver.

Similarly, for departing aircraft trajectory families $i, j \in \mathbb{D}$ we introduce the two constraints

$$t_i^F - t_i^S - \delta_{min} \geq 0 \tag{4}$$
$$t_j^F - t_j^S - \delta_{min} \geq 0 \tag{5}$$

that ensure the push back time windows for aircraft trajectory family i and j are both larger than a predefined value δ_{min}. The value δ_{min} is the minimum acceptable push back window. For example, pilots and ramp area ground crew could find a schedule that requires aircraft to initiate push back within a time window of 5 seconds too restrictive to consistently execute. In this paper we use the value $\delta_{min} = 25[s]$ when solving for

the optimal combination of sub-windows. The correct value should be determined in conjunction by ramp area controllers and pilots.

For departing aircraft trajectory families $i, j \in \mathbb{D}$ we introduce the four constraints (6) - (9)

$$t_i^F - t_i - t_i^{F0} \leq 0 \tag{6}$$
$$t_i^S - t_i - t_i^{S0} \geq 0 \tag{7}$$
$$t_j^F - t_j - t_j^{F0} \leq 0 \tag{8}$$
$$t_j^S - t_j - t_j^{S0} \geq 0 \tag{9}$$

where t_i is the target movement area time of aircraft trajectory family i and t_i^{S0} and t_i^{F0} are the earliest and latest feasible push back times for aircraft i such that the scheduled target movement area time $t_i = 0$ is enforced. The same definitions apply to the variables for aircraft trajectory family j.

To ensure that for any given combination of target movement area time schedules, given by t_i and t_j, the start and finish of the push back sub-windows defined by t^S and t^F must be within the bounds defined by the start and finish of the feasible push back window. This implies that there exists a feasible trajectory from family i and j that meets the scheduled target movement area times t_i and t_j without accounting for conflicts. These four constraints describe that the push back time windows that we solve for, which are illustrated in black solid (dotted) lines in Fig. 1b, are proper sub-windows of the original green rectangle.

For each conflict point $\kappa = 1, 2, ..., N$ we generate the set of five constraints

$$v_\kappa z_{\kappa 1}\left[t_i^F - PB^i\right] \leq 0 \tag{10}$$
$$v_\kappa z_{\kappa 2}\left[t_i^S - PB^i\right] \geq 0 \tag{11}$$
$$v_\kappa z_{\kappa 3}\left[t_j^F - PB^j\right] \leq 0 \tag{12}$$
$$v_\kappa z_{\kappa 4}\left[t_j^S - PB^j\right] \geq 0 \tag{13}$$
$$z_{\kappa 1} + z_{\kappa 2} + z_{\kappa 3} + z_{\kappa 4} = 1 \tag{14}$$

where v_κ is a binary variable associated with the conflict point κ. It is one if the constraints are valid and the point is to be outside the time window, zero otherwise. The variables $z_{\kappa 1}$, $z_{\kappa 2}$, $z_{\kappa 3}$ and $z_{\kappa 4}$ are binary variables associated with the conflict point κ that are one if the time window is to be constrained below, above, left or right of the conflict point, zero otherwise. Therefore, for any individual constraint (10)-(13) to be valid, both binary variables v_κ and z_κ associated with the constraint must be equal to one. Otherwise, the constraint will be automatically satisfied because the left hand side of the equation will evaluate to zero.

This implies that for any conflict point κ, we can set the value of v_κ to zero and automatically satisfy constraints (10)-(13). This ensures that the optimal solution will *not* constrain the conflict point κ to be outside the optimal time window. Next, we introduce the constraint

$$\Sigma_{\kappa=1}^{N} v_\kappa = N - p \tag{15}$$

where p is the number of conflict points not constrained to be outside the optimal time window. By constraining the summation of the valid bits, we ensure that the number of conflict points that are assigned valid constraints are equal to the value $N - p$.

The four nonlinear constraints (10)-(13) can be linearized [18, 19] as

$$t_i^F - PB^i - (1 - z_{\kappa 1})S - (1 - v_\kappa)S \leq 0 \tag{16}$$
$$t_i^S - PB^i + (1 - z_{\kappa 2})S + (1 - v_\kappa)S \geq 0 \tag{17}$$
$$t_j^F - PB^j - (1 - z_{\kappa 3})S - (1 - v_\kappa)S \leq 0 \tag{18}$$
$$t_j^S - PB^j + (1 - z_{\kappa 4})S + (1 - v_\kappa)S \geq 0 \tag{19}$$

where the value of S is sufficiently large. When we linearize the constraints, the value of S should be chosen to generate a constraint that is automatically satisfied for any optimal solution. For instance, if the feasible push back window is defined within the range $[0, 100]$, then generating the constraint that the start of the window should be greater than -10 is automatically satisfied by any optimal solution.

As an example, consider the expression (16) for aircraft i

$$t_i^F - PB^i - (1 - z_{\kappa 1})S - (1 - v_\kappa)S \leq 0$$

If we fix the value $z_{\kappa 1} = 1$ and $v_\kappa = 0$ the constraint simplifies to

$$t_i^F \leq PB^i + S \tag{20}$$

This constraint should be automatically satisfied by any optimal solution given $v_\kappa = 0$. For every aircraft i and any conflict point κ, the push back time that generates a conflict will be realized within the domain $PB^i \in [t_i + t_i^{S0}, t_i + t_i^{F0}]$. The worst case for the less than or equal constraint is to realize the lower bound of $PB^i = t_i + t_i^{S0}$. We use the value $S = (t_i^{F0} - t_i^{S0} + B)$ and plug in the lower bound realization of the push back time into expression (20) to get

$$t_i^F \leq t_i + t_i^{F0} + B$$

Given $B \geq 0$ the constraint is automatically satisfied by expression (6). Similar reasoning can be applied to show the constraints are automatically satisfied when $z_{\kappa 1} = 1$ and $v_\kappa = 0$ or when $z_{\kappa 1} = 0$ and $v_\kappa = 1$. Next, consider the expression

$$t_i^S - PB^i + (1 - z_{\kappa 2})S + (1 - v_\kappa)S \geq 0$$

If we fix the value $z_{\kappa 2} = 1$ and $v_\kappa = 0$ the constraint simplifies to

$$t_i^S \geq PB^i - S$$

substituting for $S = (t_i^{F0} - t_i^{S0} + B)$ and the worse case $PB^i = t_i^{F0}$ for the greater than or equal to constraint we get

$$t_i^S \geq t_i + t_i^{S0} - B$$

Given $B \geq 0$ the constraint is automatically satisfied by expression (7). Similar reasoning can be applied to show the constraints are automatically satisfied when $z_{\kappa 2} = 1$ and $v_\kappa = 0$ or when $z_{\kappa 2} = 0$ and $v_\kappa = 1$.

IV. Analysis of MILP for Optimal Chance-Constrained Time Windows

The MILP can solve for the optimal chance-constrained time windows given any distribution of conflict points. In this paper we analyze two sample problems that are qualitatively different. The distribution of conflict points are selected to analyze the performance of the algorithm and are not representative of sampled conflict distribution from our stochastic model.

The first sample domain can be seen in Fig. 2a - Fig. 2c where we provide solutions that allow zero, three, and ten conflict points inside the time window. We call this domain the *easy* domain since there is only one main cluster of conflict points. Aside from the main conflict cluster, there are several rare samples within the otherwise empty domain.

The second domain that we analyze in this paper can be seen in Fig. 2d - Fig. 2f where we provide solutions that allow zero, three, and ten conflict points inside the time window. This domain we define as the *hard* domain. This domain is difficult because there exist symmetries that could produce near optimal solutions subject to chance. Aside from the two main conflict clusters, we introduce a couple of rare samples within the otherwise empty domain.

As can be seen in Fig. 2, the quality of solution can be dramatically impacted by the few samples that we introduce into the otherwise empty domain. Particularly we can see that in Fig. 2a the solution is affected by the presence of three conflict points. By allowing these conflict points inside the time window the solution becomes much more appealing. These rare conflict points within the time window would likely be resolved by pilots slowing down or stopping along the route. This introduces an intriguing trade-off.

The solutions are influenced by the choice of parameter ϵ that appears in objective function 1. In particular, setting $\epsilon = 0$ solves for the optimal time window that is defined to maximize the minimal edge. Setting $\epsilon = 1$ solves for the time window that maximizes the summation of push back time windows (perimeter of time window). Any value $\epsilon \in [0, 1]$ can also be selected allowing us to mix the two objectives.

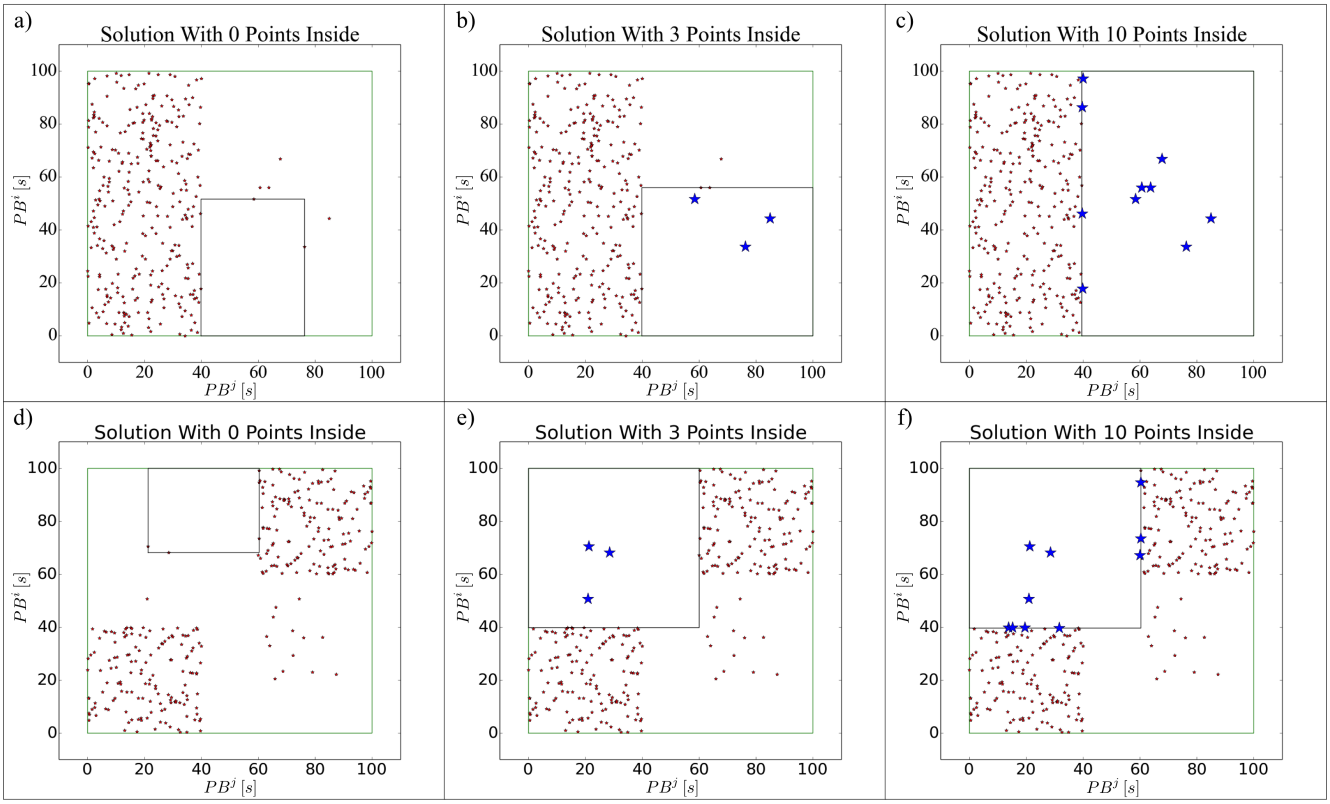

Figure 2. Figure 2a - Fig. 2c show the optimal time window solution on the "easy" domain allowing 0, 3, and 10 points inside the time window respectively. The red (blue) points are color coded to illustrate which points are outside (inside) the optimal time window. Figure 2d - Fig. 2f show the optimal time window solution on the "hard" domain allowing 0, 3, and 10 points inside the time window respectively. The red (blue) points are color coded to illustrate which points are outside (inside) the optimal time window.

A. Runtime of MILP

The runtime of the Gurobi Optimizer [20] solver is influenced by the distribution of the red conflict points. Figure 3a illustrates that both objectives can be efficiently solved on the easy domain. For the "easy domain", the maximum min edge computation ($\epsilon = 0$) is done faster than the maximum perimeter ($\epsilon = 1$) computation time on average, but the outperformance is not dramatic. For the "hard domain," the solver is not able to efficiently solve the maximum perimeter objective when we allow 30 points inside the time window. Allowing only 10 points inside the time window can take up to 100,000 seconds for the solver to return an answer. Because of this, we omit the maximum perimeter computation time on the hard domain in Fig.. 3a so that we do not lose perspective.

Figure 3b and Fig. 3c illustrate the sensitivity of the runtime to the selection of parameter ϵ in the objective function (1). Figure 3b shows the average runtime in solid blue of the min edge objective solving on the "hard domain" for 20 random inputs, allowing $p = 0, 1, 2, ..., 50$ conflict points inside the time window. The dotted blue lines represent the average computation time plus or minus one standard deviation. Figure 3c shows the average runtime in solid blue of the perimeter objective solving on the "hard domain" for 20 random inputs, allowing $p = 0, 1, 2, ..., 5$ conflict points inside the time window. The dotted blue lines represent the average computation time plus or minus one standard deviation.

The runtime of the algorithm is also influenced by the number of points allowed inside the time window. Given N conflict points, and allowing p inside the time window, the number of combinatorial possibilities we must consider is given by $\binom{N}{p}$. As the number of points p inside the time window increases, the computational complexity of the problem increases. Figure 3b and Fig. 3c illustrate the increase in runtime as the number of points inside the time window increases. The increased difficulty of the problem is dramatic in Fig. 3c where increasing the number of conflict points inside the time window from two to five increased the average runtime of the algorithm from less than 100 seconds to 1000 seconds.

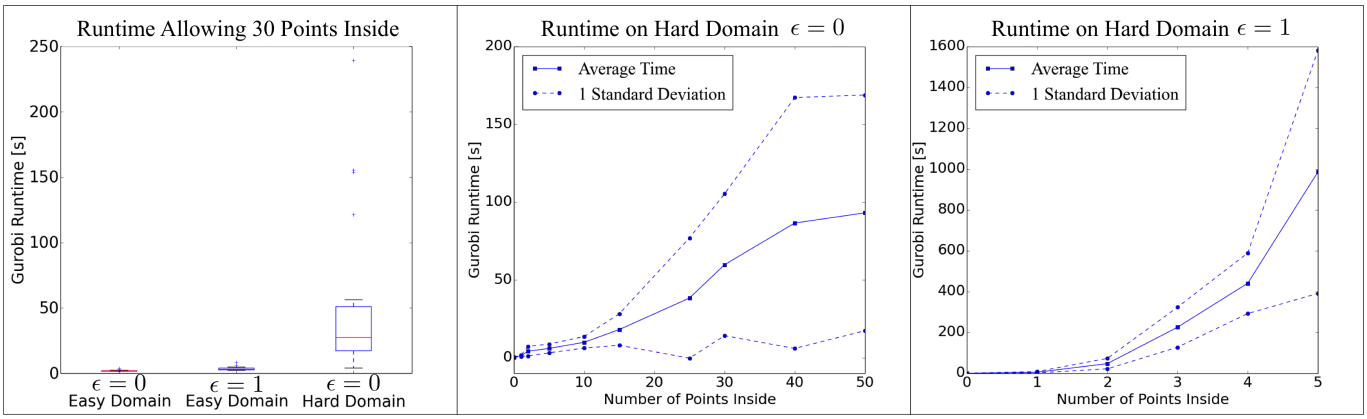

Figure 3. a) Runtime of the Gurobi solver for different objective function applied to the easy and hard problems allowing 30 conflict points inside the window. The runtime of the maximum perimeter objective applied to the hard domain is omitted as it can take up to 100,000 [s] to execute. b) The average runtime is plotted in solid blue of the min edge objective solving on the "hard domain" for 20 random inputs, allowing $p = 0, 1, 2, ..., 50$ conflict points inside the time window. The dotted blue lines represent the average computation time plus or minus one standard deviation. c) The average runtime is plotted in solid blue of the perimeter objective solving on the "hard domain" for 20 random inputs, allowing $p = 0, 1, 2, ..., 5$ conflict points inside the time window. The dotted blue lines represent the average computation time plus or minus one standard deviation.

Algorithm 1 Cascading MILP Cut: Constrain Window Right ($z_{\kappa 4}$)

for $i = 2$:N **do**
  $j = i - 1$
  $z_{i4} \geq z_{j4}$
end for

B. Improving the Runtime of the MILP Approach

The performance of the MILP can be improved by enforcing cutting planes [13, 21–24] to the solution space. A cutting plane is a valid inequality that improves the linear relaxation of the problem to more closely approximate the integer programming problem. This topic is important because improving formulations with cutting planes is of interest independently of the algorithm used to solve the problem [24]. A particularly interesting algorithm is the branch-and-cut method where the cutting plane method improves the relaxation of the problem, and branch-and-bound algorithms proceed by a sophisticated divide and conquer approach to solve the problem [22].

Figure 4a shows four red conflict points and a blue time window. The conflict points have been labeled in decreasing order of their x-coordinate using the black labels. Furthermore, we imagine a situation in which the constraint that enforces the time window should be to the right of $\kappa = 1$ has been activated, i.e. $z_{14} = 1$. Given the ordering of conflict points in decreasing order, we can immediately generate a set of linear constraints that define cutting planes. Every conflict point to the left of conflict point $\kappa = 1$ is by definition also to the left of the time window. Therefore, we can enforce $z_{\kappa 4} \geq 1$ for all conflict points $\kappa > 1$ and cut the feasible solution space. If we were to apply this cutting method we would generate $O(N^2)$ additional constraints, looping through every conflict point twice.

Instead, consider implementing the cut: if the time window is to the right of conflict point $\kappa = 1$, then the time window is also to the right of conflict point $\kappa = 2$. In algebriac terms this cut takes the form $z_{24} \geq z_{14}$. Next, apply the same logic to the conflict point $\kappa = 2$. Iterating through, the loop eventually hits the left most conflict point and every conflict point to the left of conflict point $\kappa = 1$ is set with binary variable $z_{\kappa 4} = 1$. Given that conflict point $\kappa = 1$ is to the left of the time window and the ordering of the conflict points in descending values of x coordinate; setting the binary variable $z_{\kappa 4} = 1$ for the conflict points $\kappa = 2, 3, 4$ seen in Fig. 4a would satisfy the system of constraints (14, 16-19). This cuts the feasible solution space while using only $O(N)$ constraints, see Algorithm 1. Notice the value of $j = i - 1$ fixes the binary variable $z_{\kappa 4}$ for only the adjacent conflict point as opposed to looping through the indexes $j = 1, 2, ..., i - 1$.

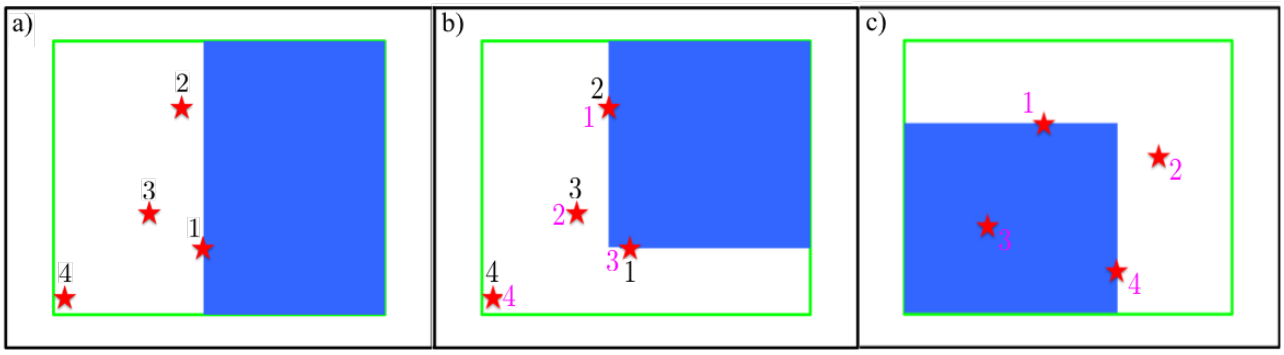

Figure 4. a) Conflict point 1 will activate the cascading constraints which ensure the time window is constrained to the right of conflict point 2 ($z_{24} = 1$), 3 ($z_{34} = 1$), and 4 ($z_{44} = 1$). b) The time window will be constrained to be *both* above conflict point $\kappa = 4$ (magenta) and to the right of conflict point $\kappa = 4$ (black). This violates constraint (14) and the solution is no longer feasible without a modification using constraints (21 - 22) instead. c) If the solver assigns $z_{\hat{\kappa}2} = 1$ for the conflict point $\hat{\kappa} = 3$ that is inside the time window. This implies that the constraints enforce the time window to be above the conflict point $\hat{\kappa} = 3$. Applying cascading constraints to constrain the window above $\hat{\kappa} = 3$ would enforce the time window to also be above conflict point $\kappa = 4$ because the constraints will enforce $z_{42} >= z_{32}$. The solver will then either assign the valid bit $v_4 = 0$, or constrain the time window to be above the conflict point 4, both of which are undesired.

Algorithm 2 Cascading MILP Cuts: Constrain Window Above ($z_{\kappa 2}$) And Right ($z_{\kappa 4}$)

for $i = 2$:N **do**
  $j = i - 1$
  $z_{i2} \geq z_{j2}$
  $z_{i4} \geq z_{j4}$
end for

Algorithm 3 Cascading MILP Cuts: Constrain Window All Directions

for $i = 2$:N **do**
  $j = i - 1$
  $1 + z_{i1} \geq z_{j1} + v_j$
  $1 + z_{i2} \geq z_{j2} + v_j$
  $1 + z_{i3} \geq z_{j3} + v_j$
  $1 + z_{i4} \geq z_{j4} + v_j$
end for

We can apply cuts to the solution space in orthogonal directions at the same time. Figure 4b shows a time window that has been constrained above the magenta conflict point $\kappa = 3$ (vertical ordering labeled with magenta) and to the right of the black conflict point $\kappa = 2$ (horizontal ordering labeled with black). The time window will be constrained to the right of the black conflict point 3 enforced by the black conflict point 2 ($z_{34} \geq z_{24}$ for black κ) , and the time window will be constrained to the right of the black conflict point 4 enforced by the black conflict point 3 ($z_{44} \geq z_{34}$ for black κ). The time window will also be constrained to be above the magenta conflict point 4 enforced by the magenta conflict point 3 ($z_{42} \geq z_{32}$ for magenta κ). The cuts as described above can be implemented by the algebraic constraints seen in Algorithm 2.

Enforcing cuts at the same time in orthogonal directions will lead to unfeasible solutions. An example is conflict point 4 (labeled in both magenta and black) located in the lower left hand corner of the domain of Fig. 4b. The Algorithm 2 will enforce $z_{42} = 1$ *and* $z_{44} = 1$; this is because the conflict point $\kappa = 4$ is indeed *both* below and to the left of the time window. When the value of both binary variables are set to 1, we violate constraint (14) and the solution is no longer feasible.

This problem can be addressed by replacing constraint (14) with the two constraints

$$z_{\kappa 1} + z_{\kappa 2} + z_{\kappa 3} + z_{\kappa 4} \geq 1 \tag{21}$$
$$z_{\kappa 1} + z_{\kappa 2} + z_{\kappa 3} + z_{\kappa 4} \leq 2 \tag{22}$$

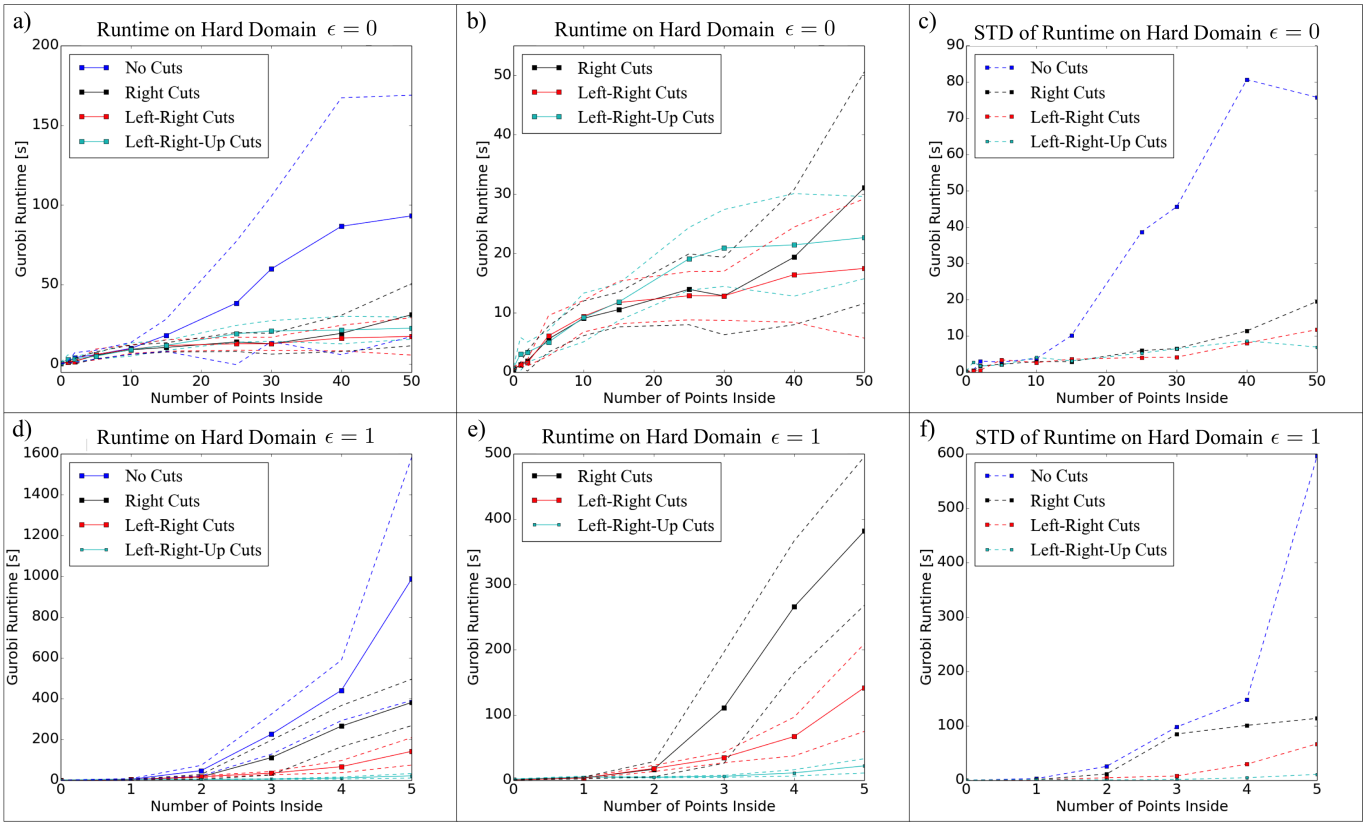

Figure 5. In the top row Fig. 5a - Fig.5c we report the results for the min edge objective and the bottom row Fig. 5d - Fig.5f we report the results for the maximum perimeter objective. In the first column we plot the average runtime of the various computation methods in solid colors and plot the average runtime plus or minus the standard deviation in dotted lines. In the second column we plot the average runtime of the cutting methods only in solid colors and plot the average runtime plus or minus the standard deviation in dotted lines. In the third column we plot the standard deviation of the various methods.

which allow for any given conflict point to be assigned to two orthogonal directions in relation to the time window at the same time. Applying Algorithm 2 to the conflict points in Fig. 4b will satisfy the system of constraints (16-19, 21-22)

To enforce cuts in all directions at the same time, a modification must be made to the cutting algorithm. In Algorithm 1 anytime the binary variable $z_{j4} = 1$, the binary variable $z_{i4} = 1$ was enforced, and this generated a cut to the solution space. This cut is generated even if the valid bit $v_\kappa = 0$. When we generate cuts in all directions, a conflict point $\hat{\kappa}$ that is allowed inside the time window will be assigned a $false$ value for either $z_{\hat{\kappa}1}$, $z_{\hat{\kappa}2}$, $z_{\hat{\kappa}3}$, or $z_{\hat{\kappa}4}$. Since we are cutting in all directions, any of these false assignments will generate unwanted cuts to conflict points outside the time window. If we were not cutting in all directions, the conflict point $\hat{\kappa}$ that is inside the time window can assign $z_{\hat{\kappa}1}$, $z_{\hat{\kappa}2}$, $z_{\hat{\kappa}3}$, or $z_{\hat{\kappa}4}$ equal to 1 in the direction that will not be cutting the solution space. This ensures that no unwanted cuts are generated which could eliminate feasible solutions.

For example, see Fig. 4c where the conflict point $\hat{\kappa} = 3$ is inside the time window. To satisfy constraints (21-22) the solver must assign the value one to any of the binary variables $z_{\hat{\kappa}1}$, $z_{\hat{\kappa}2}$, $z_{\hat{\kappa}3}$ or $z_{\hat{\kappa}4}$. Imagine the solver has selected the value $z_{\hat{\kappa}2} = 1$. This implies that the constraints enforce the time window to be above the conflict point $\hat{\kappa} = 3$. Applying cascading constraints to constrain the window above $\hat{\kappa} = 3$ would enforce the time window to also be above conflict point $\kappa = 4$ because the constraints will enforce $z_{42} >= z_{32}$. The solver will then constrain the time window to be above the conflict point 4 or assign the valid bit $v_4 = 0$, both of which are unwanted solutions.

We can address this issue by modifying the algorithm that cuts our solution space. Algorithm 3 shows the adjusted cutting scheme. Instead of enforcing the constraint $z_{42} \geq= z_{32}$ we now enforce the constraint $1 + z_{42} \geq z_{32} + v_3$. This implies that the binary variable z_{42} associated with conflict point $\kappa = 4$ is only

turned on if $z_{32} = 1$ *and* the valid bit associated with $\kappa = 3$ is turned on with $v_3 = 1$. In the example shown in Fig. 4c, the valid bit $v_3 = 0$ will not apply cascading cuts to the conflict point $\kappa = 4$, and we do not constrain the time window to be above the conflict point 4 or assign the valid bit $v_4 = 0$

Figure 5a - Fig.5f which shows the average runtime of the various cutting methods. In the top row Fig. 5a - Fig. 5c report the results for the min edge objective ($\epsilon = 0$) and the bottom row Figure 5d - Fig.5f report the results for the maximum perimeter objective ($\epsilon = 1$). In the first column we plot the average runtime of the various computation methods in solid colors and plot the average runtime plus or minus the standard deviation in dotted lines. In Fig 5a - Fig.5f the average runtime and the standard deviation are calculated for 20 random inputs of conflict points. In the second column we focus on the cutting methods only and plot the average runtime in solid colors and plot the average runtime plus or minus the standard deviation in dotted lines. In the third column we plot the standard deviation of the various methods.

Applying cuts to the solution space helped reduce the average runtime of both the min edge objective and the perimeter objective. As can be seen in Fig. 5a and Fig. 5d, cutting the solution space in the right, left-right, and left-right-up directions provided much better computation results than applying no cuts to the solution space. From Fig. 5b and Fig. 5e we can conclude that left-right cuts were the most efficient for the min edge objective and left-right-up cuts were the most efficient for the max perimeter objective. From Fig. 5c and Fig. 5f we can conclude that cutting the solution space not only improves the average runtime, but also reduces the standard deviation in computation time for the solver.

Cutting the solution space in all directions had a negative impact on the runtime and underperformed applying no cuts to the solution space. We do not display the results in Fig. 5a - Fig.5f so that we can focus on the cutting methods that improved the runtime. We find the slowdown of cutting in all directions to be counterintuitive. Applying cuts should reduce the size of the solution space, which eliminates non-integer solutions to the relaxation problem, and therefore help the solver. The increase in runtime could be an artifact of the modified constraints $1 + z_{42} \geq z_{32} + v_3$ as opposed to the constraints $z_{42} \geq z_{32}$.

V. Discussion and Future Work

In this work, we formulated a MILP model to solve for the optimal chance-constrained push back time windows. The time windows are chance-constrained because they allow for a non-zero but bounded probability of conflicts among the sampled aircraft trajectories. These solutions are acceptable because the conflicts may be extremely rare. Solutions of the MILP were shown to be significantly impacted by the presence of even a few conflict points within an otherwise empty domain. By allowing for some conflict points inside the time windows, the solutions become much more attractive and the trade-off between the increased size of the push back window and the small risk of conflict becomes appealing.

The main contribution of the MILP is to compute push back time windows that allow for aircraft to taxi unimpeded from their gate to the departure runway queue in the presence of other aircraft and trajectory uncertainties. This allows for ramp controllers to better manage surface traffic and reduce fuel consumption while scheduling the push back for ramp area aircraft. Ideally, the MILP will be used for real-time decision making by controllers so the computational runtime is a critical component of the tool.

The runtime of the MILP was shown to be most influenced by a parameter within the objective function, the distribution of conflict points, and the number of conflict points that are allowed inside the time window. Maximizing the minimum time window was found to be much more efficient than maximizing the perimeter of the time window. This is true even though all the constraints and formulation of the MILP is the same, the only difference is the selection of parameter ϵ within the objective function. In addition, the runtime of the algorithm was shown to be sensitive to the distribution of conflict points, and the algorithm was shown to execute much more efficiently on an "easy" domain than a "hard" domain.

In order to address the the issues with the runtime, we introduced cutting planes to cut the solution space. A cutting plane is a valid inequality that improves the linear relaxation of the problem to more closely approximate the integer programming problem. Various cutting methods were investigated and applied to the MILP. Overall, the analysis showed that the cutting methods reduced the runtime and standard deviation of the runtime for both the maximum min edge objective and the maximum perimeter objective.

Future work will consider techniques to reduce the overall execution time of the MILP. The algorithm must execute in less time if we are to implement the solutions for real-time decision making. Future work will also investigate the trade off between the number of points that are allowed inside the time window and the frequency at which pilots have to slow down or stop to avoid a loss of separation.

References

[1]Gupta, G., Malik, W., and Jung, Y. C., "A Mixed Integer Linear Program for Airport Departure Scheduling," *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, Hilton Head, South Carolina, 2009.

[2]Malik, W., Gupta, G., and Jung, Y. C., "Managing Departure Aircraft Release for Efficient Airport Surface Operations," *Proceedings of the AIAA Guidance, Navigation, and Control (GNC) Conference*, Toronto, Canada, 2010.

[3]Gupta, G., Malik, W., and Jung, Y. C., "Incorporating Active Runway Crossings in Airport Departure Scheduling," *AIAA Guidance, Navigation, and Control Conference (GNC)*, Toronto, Ontario Canada, 2010.

[4]Malik, W., Gupta, G., and Jung, Y. C., "Spot Release Planner: Efficient Solution for Detailed Airport Surface Trffic Optimization," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, Indianapolis, Indiana, 2012.

[5]Jung, Y. C., Hoang, T., Montoya, J., Gupta, G., Malik, W., and Tobias, L., "A Concept and Implementation of Optimized Operations of Airport Surface Traffic," *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Fort Worth, TX*, 2010.

[6]Gupta, G., Malik, W., and Jung, Y. C., "An Integrated Collaborative Decision Making and Tactical Advisory Concept for Airport Surface Operations Management," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Indianapolis, IN*, 2012.

[7]Gupta, G., Malik, W., Tobias, L., Jung, Y., Hoang, T., and Hayashi, M., "Performance Evaluation of Individual Aircraft Based Advisory Concept for Surface Management," *Tenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2013)*, Chicago, Il, 2013.

[8]Nikoleris, T., Gupta, G., and Kistler, M., "Detailed Estimation of Fuel Consumption and Emissions During Aircraft Taxi Operations at Dallas Fort Worth International Airport," *Published in the journal Transportation Research Part D: Transport and Environment, Vol. 16D, Issue 4*, June 2011.

[9]Jung, Y., Hoang, T., Montoya, J., Gupta, G., Malik, W., L., T., and Wang, H., "Performance Evaluation of a Surface Traffic Management Tool for Dallas/Fort Worth International Airport," *9th USA/Europe Air Traffic Management Research and Development Seminar*, 2011, pp. 1–10.

[10]Coupe, W. J., Milutinović, D., Malik, W., Gupta, G., and Jung, Y., "Robot Experiment Analysis of Airport Ramp Area Time Constraints," *AIAA Guidance, Navigation, and Control Conference (GNC)*, Boston, MA, 2013.

[11]Coupe, W. J., Milutinović, D., Malik, W., and Jung, Y., "Integration of Uncertain Ramp Area Aircraft Trajectories and Generation of Optimal Taxiway Schedules at Charlotte Douglas (CLT) Airport," *AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, Dallas, TX, 2015.

[12]Wolsey, L. A. and Nemhauser, G. L., *Integer and Combinatorial Optimization*, John Wiley & Sons, 2014.

[13]Nemhauser, G. L., "The Age of Optimization: Solving Large-scale Real-world Problems," *Operations Research*, Vol. 42, No. 1, 1994, pp. 5–13.

[14]Klotz, E. and Newman, A. M., "Practical Guidelines for Solving Difficult Mixed Integer Linear Programs," *Surveys in Operations Research and Management Science*, Vol. 18, No. 1, 2013, pp. 18–32.

[15]Coupe, W. J., Milutinović, D., Malik, W., and Jung, Y., "Optimization of Push Back Time Windows That Ensure Conflict Free Ramp Area Aircraft Trajectories," *AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, Dallas, TX, 2015.

[16]Bertsekas, D. P., "Nonlinear Programming," 1999.

[17]Charnes, A. and Cooper, W. W., "Chance-Constrained Programming," *Management science*, Vol. 6, No. 1, 1959, pp. 73–79.

[18]Glover, F., "Improved Linear Integer Programming Formulations of Nonlinear Integer Problems," *Management Science*, Vol. 22, No. 4, 1975, pp. 455–460.

[19]Adams, W. P. and Forrester, R. J., "Linear Forms of Nonlinear Expressions," *Operations Research Letters*, Vol. 35, No. 4, 2007, pp. 510–518.

[20]Gurobi Optimization, Inc., "Gurobi Optimizer Reference Manual," 2015.

[21]Padberg, M. W., "Classical Cuts for Mixed-Integer Programming and Branch-and-Cut," *Mathematical Methods of Operations Research*, Vol. 53, 2001.

[22]Mitchell, J. E., "Branch-and-cut Algorithms for Combinatorial Optimization Problems," *Handbook of applied optimization*, 2002, pp. 65–77.

[23]Cornuéjols, G., "Valid Inequalities for Mixed Integer Linear Programs," *Mathematical Programming*, Vol. 112, No. 1, 2008, pp. 3–44.

[24]Marchand, H., Martin, A., Weismantel, R., and Wolsey, L. A., "Cutting Planes in Integer and Mixed Integer Programming," *Discrete Applied Mathematics*, Vol. 123, No. 1-3, 2002, pp. 397–446.